

LAPORAN TUGAS KECIL I

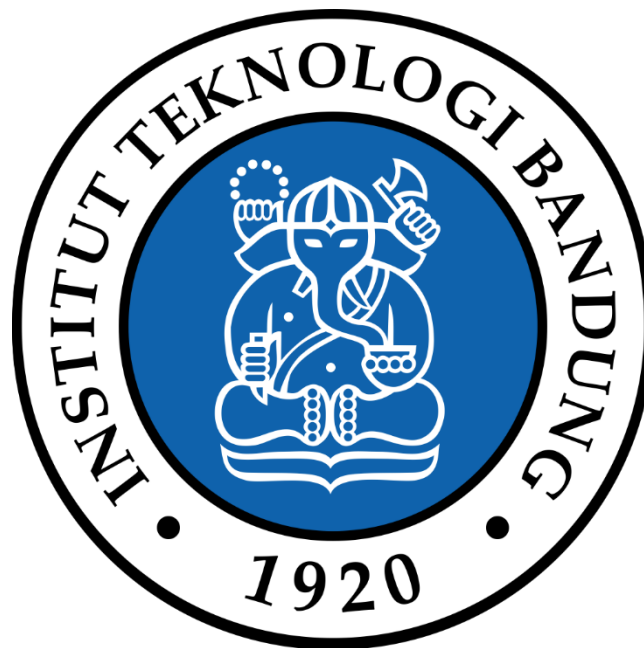
“Penyelesaian Word Search Puzzle dengan Algoritma Brute Force”

Laporan Ini Dibuat Untuk Memenuhi Tugas Perkuliahan

Mata Kuliah Strategi Algoritma (IF2211)

KELAS 02

Dosen : Dr. Nur Ulfa Maulidevi, S.T., M.Sc.



DISUSUN OLEH:

Rava Naufal Attar (13520077)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2021/2022

Daftar Isi

Daftar Isi	2
Algoritma Brute Force	3
Source Program	6
Screenshot	13
Test Case 1	13
Test Case 2	16
Test Case 3	20
Test Case 4	23
Test Case 5	28
Test Case 6	32
Test Case 7	38
Test Case 8	50
Alamat Drive	63
<i>Lampiran</i>	64

Algoritma Brute Force

Pertama dilakukan pembacaan file dari file.txt, kemudian data dari file tersebut akan diolah dan dipisahkan menjadi dua bagian yang berbeda.

```
char[][] puzzle;  
String[] keys_arr;
```

Sebagian besar program dikontrol oleh variabel,

```
Boolean found;  
Int[] keys_stat;
```

yang mana *variabel found* akan bernilai *True* apabila semua keyword dari puzzle telah ditemukan. Sedangkan untuk *array keys_stat* menyatakan status dari setiap keyword yang terdiri dari 0/1 yang merepresentasikan keyword ke-i telah ditemukan apabila $keys_stat[i] = 1$.

Program akan mengiterasi setiap elemen dari matriks puzzle disetiap pencarian kata. Kemudian disetiap iterasi elemen tersebut, akan dilakukan pencarian dengan referensi arah mata angin. Yang mana pada kode program direpresentasikan sebagai berikut.

```
0 : Utara  
1 : Timur Laut  
2 : Timur  
3 : Tenggara  
4 : Selatan  
5 : Barat Daya  
6 : Barat  
7 : Barat Laut
```

Pada setiap pencarian kata tersebut, apabila kata telah ditemukan maka program akan menampilkan kata, matriks huruf dari kata yang ditemukan tersebut, dan jumlah perbandingan huruf yang dilakukan untuk menemukan kata tersebut.

Pencarian kata akan selesai apabila *array keys_stat* telah terisi dengan angka 1 pada setiap elemennya, yang berarti setiap keyword telah ditemukan.

Variables	Keterangan
<code>char[][] puzzle</code>	Matriks huruf dari puzzle
<code>int puzzle_rows</code>	Jumlah kolom matriks puzzle
<code>int puzzle_cols</code>	Jumlah baris matriks puzzle
<code>String[] keys_arr</code>	List dari keywords yang ingin dicari
<code>int[] keys_stat</code>	Status dari setiap keywords yang ada
<code>int[] key_comp</code>	Jumlah perbandingan huruf yang dilakukan untuk menemukan kata didalam puzzle
<code>boolean found</code>	Menyatakan apakah seluruh keyword telah ditemukan atau belum
<code>int key_count</code>	Sebagai iterasi untuk setiap keyword yang ada
<code>int row</code>	Sebagai iterasi baris matriks puzzle
<code>int col</code>	Sebagai iterasi kolom matriks puzzle
<code>int dir_state</code>	Sebagai iterasi pada saat pencocokan huruf dengan referensi arah mata angin
<code>int boundary_length</code>	Panjang batasan dalam matriks puzzle
<code>int x_boundary</code>	Batas horizontal dalam matriks puzzle
<code>int y_boundary</code>	Batas vertikal dalam matriks puzzle
<code>int finding_idx</code>	Sebagai pencatat indeks pada saat pencocokan huruf
<code>String curr_key</code>	Sebagai penunjuk keyword yang sedang di cek
<code>long startTime</code>	Sebagai pencatat waktu eksekusi program

Methods	Keterangan
<code>public static void displayMenu()</code>	Untuk menampilkan layar awal
<code>public static char[][] readPuzzle(String file)</code>	Untuk membuka file dan mengembalikan matrix dari puzzle
<code>public static String[] readKeywords(String file)</code>	Untuk membuka file dan mengembalikan array dari keywords
<code>public static char getCharFromString(String str, int index)</code>	Untuk mendapatkan char dari suatu string
<code>public static void fillKeystat(int[] keys_stat)</code>	Untuk menginisialisasi array status dari key sama dengan nol
<code>public static int[][] clearArr_2d(int[][] array)</code>	Untuk mengosongkan array 2D
<code>public static boolean isRowCol_avail(int row, int col, int[][] keys_constructors)</code>	Untuk mengecek keberadaan suatu titik {row, col} pada array 2D dari posisi keyword yang ditemukan
<code>public static boolean isAllOne(int[] array)</code>	Untuk mengecek apakah semua elemen pada array bernilai sama dengan satu
<code>public static void print_key_found(char[][] puzzle, String key_word, int key_comp, int[][] key_const)</code>	Untuk menampilkan keywords yang ditemukan, posisinya pada puzzle, serta jumlah perbandingan huruf yang dilakukan

Source Program

```
1 package src;
2
3 import java.io.*;
4 import java.util.*;
5
6 public class Main {
7     static Scanner in = new Scanner(System.in);
8
9     // Method untuk menampilkan layar awal
10    public static void displayMenu() {
11        System.out.println("=====");
12        System.out.println("    (^///^) SELAMAT DATANG (^///^)    ");
13        System.out.println("=====");
14        System.out.println("Masukkan nama file (xxx.txt): ");
15        System.out.print("> ");
16    }
17
18    // Method membuka file dan mengembalikan matrix dari puzzle
19    public static char[][] readPuzzle(String file) throws IOException {
20        char[][] puzzle = null;
21        String puzzle_item = "";
22
23        try {
24            BufferedReader reader = new BufferedReader(new FileReader(file));
25            String line = reader.readLine();
26            if (line != null) {
27                while (line != null && line.length() > 0) {
28                    puzzle_item += line.toUpperCase();
29                    puzzle_item += "\n";
30                    line = reader.readLine();
31                }
32                String[] puzzle_row = puzzle_item.split("\n");
33                int rows = puzzle_row.length;
34                String[] puzzle_col = puzzle_row[0].split(" ");
35                int cols = puzzle_col.length;
36
37                puzzle = new char[rows][cols];
38                for(int i = 0; i < rows; i++) {
39                    String str = puzzle_row[i].replaceAll("\s", "");
40                    char[] array = str.toCharArray();
41                    for(int j = 0; j < cols; j++){
42                        puzzle[i][j] = array[j];
43                    }
44                    // puzzle[i] = str.toCharArray();
45                }
46            }
47        }
```

```

47         return puzzle;
48     }
49     catch (FileNotFoundException e) {}
50     return null;
51 }
52
53 // Method membuka file dan mengembalikan array dari keywords
54 public static String[] readKeywords(String file) throws IOException {
55     String[] key_words = null;
56     String key_word = "";
57
58     try {
59         BufferedReader reader = new BufferedReader(new FileReader(file));
60         String line = reader.readLine();
61         if (line != null) {
62             while (line != null && line.length() > 0) {
63                 line = reader.readLine();
64             }
65             line = reader.readLine();
66
67             while (line != null) {
68                 key_word += line.toUpperCase();
69                 key_word += "\n";
70                 line = reader.readLine();
71             }
72             key_words = key_word.split("\n");
73         }
74         return key_words;
75     }
76     catch (FileNotFoundException e) {}
77     return null;
78 }
79
80 // Method untuk mendapatkan char dari suatu string
81 public static char getCharFromString(String str, int index) {
82     return str.charAt(index);
83 }
84
85 // Method untuk menginisialisasi array status dari key = 0
86 public static void fillKeystat(int[] keys_stat) {
87     for(int i = 0; i < keys_stat.length; i++){
88         keys_stat[i] = 0;
89     }
90 }
91
92 // Method untuk mengosongkan array 2D

```

```

93     public static int[][] clearArr_2d(int[][] array) {
94         int[][] newArr = new int[0][0];
95
96         return newArr;
97     }
98
99     // Method untuk mengecek keberadaan suatu titik {row, col} pada array 2D keys_constructors
100    public static boolean isRowCol_avail(int row, int col, int[][] keys_constructors) {
101        int[] point = {row, col};
102        boolean found = false;
103
104        for(int i = 0; i < keys_constructors.length; i++) {
105            if(point[0] == keys_constructors[i][0] && point[1] == keys_constructors[i][1]) {
106                found = true;
107                break;
108            }
109        }
110        return found;
111    }
112
113    // Method untuk mengecek apakah semua elemen pada array bernilai sama dengan satu
114    public static boolean isAllOne(int[] array) {
115        for(int i = 0; i < array.length; i++) {
116            if (array[i] == 0) {
117                return false;
118            }
119        }
120        return true;
121    }
122
123    // Method untuk menampilkan keywords yang ditemukan,
124    // posisinya pada puzzle, serta jumlah perbandingan huruf yang dilakukan
125    public static void print_key_found(char[][] puzzle, String key_word, int key_comp, int[][] key_const) {
126        System.out.println(key_word);
127        for (int i = 0; i < puzzle.length; i++) {
128            for (int j = 0; j < puzzle[0].length; j++) {
129                if (isRowCol_avail(i, j, key_const)) {
130                    System.out.print(puzzle[i][j]);
131                    System.out.print(" ");
132                } else {
133                    System.out.print("-");
134                    System.out.print(" ");
135                }
136            }
137            System.out.println();
138        }

```



```

139     System.out.println("Total perbandingan huruf: " + key_comp);
140     System.out.println();
141 }
142
Run | Debug
143 public static void main(String[] args) throws IOException {
144     String inputFile;
145     displayMenu();
146     inputFile = in.nextLine();
147     System.out.println("=====");
148
149     char[][] puzzle = readPuzzle("test/" + inputFile);
150     int puzzle_rows = puzzle.length;
151     int puzzle_cols = puzzle[0].length;
152
153     String[] keys_arr = readKeywords("test/" + inputFile);
154     int[] keys_stat = new int[keys_arr.length];
155     int[] key_comp = new int[keys_arr.length];
156
157     boolean found = false;
158     int key_count = 0;
159     int row = 0;
160     int col = 0;
161     int dir_state = 0;
162
163     int boundary_length;
164     int x_boundary;
165     int y_boundary;
166     int finding_idx;
167     String curr_key;
168
169     long startTime = System.nanoTime();
170
171     while(!found && key_count < keys_arr.length) {
172         curr_key = keys_arr[key_count];
173
174         row = 0;
175         while(row < puzzle_rows && keys_stat[key_count] == 0) {
176             col = 0;
177             while(col < puzzle_cols && keys_stat[key_count] == 0) {
178                 dir_state = 0;
179                 while(dir_state < 8 && keys_stat[key_count] == 0) {
180                     int[][] key_const = new int[curr_key.length()][2];
181                     if (dir_state == 0) {
182                         boundary_length = row + 1;
183

```

```

183
184         if(curr_key.length() <= boundary_length) {
185             finding_idx = 0;
186
187             while ((finding_idx < curr_key.length()) && (getCharFromString(curr_key, finding_idx) == puzzle[row-finding_idx][col])) {
188                 int[] key_idx = new int[2];
189                 key_idx[0] = row-finding_idx;
190                 key_idx[1] = col;
191                 key_const[finding_idx] = key_idx;
192                 key_comp[key_count]++;
193                 finding_idx++;
194             }
195             if (finding_idx == curr_key.length()) {
196                 keys_stat[key_count] = 1;
197                 print_key_found(puzzle, curr_key, key_comp[key_count], key_const);
198                 key_const = clearArr_2d(key_const);
199             } else {
200                 key_const = clearArr_2d(key_const);
201             }
202         }
203     }
204     else if (dir_state == 1) {
205         boundary_length = row + 1;
206         x_boundary = puzzle_cols - col;
207
208         if (x_boundary < boundary_length) {
209             boundary_length = x_boundary;
210         }
211         if(curr_key.length() <= boundary_length) {
212             finding_idx = 0;
213
214             while ((finding_idx < curr_key.length()) && (getCharFromString(curr_key, finding_idx) == puzzle[row-finding_idx][col+finding_idx])) {
215                 int[] key_idx = new int[2];
216                 key_idx[0] = row-finding_idx;
217                 key_idx[1] = col+finding_idx;
218                 key_const[finding_idx] = key_idx;
219                 key_comp[key_count]++;
220                 finding_idx++;
221             }
222             if (finding_idx == curr_key.length()) {
223                 keys_stat[key_count] = 1;
224                 print_key_found(puzzle, curr_key, key_comp[key_count], key_const);
225                 key_const = clearArr_2d(key_const);
226             } else {
227                 key_const = clearArr_2d(key_const);
228             }
229         }
230     }

```

```

230
231     else if (dir_state == 2) {
232         boundary_length = puzzle_cols - col;
233
234         if(curr_key.length() <= boundary_length) {
235             finding_idx = 0;
236
237             while ((finding_idx < curr_key.length()) && (getCharFromString(curr_key, finding_idx) == puzzle[row][col+finding_idx])) {
238                 int[] key_idx = new int[2];
239                 key_idx[0] = row;
240                 key_idx[1] = col+finding_idx;
241                 key_const[finding_idx] = key_idx;
242                 key_comp[key_count]++;
243                 finding_idx++;
244             }
245             if (finding_idx == curr_key.length()) {
246                 keys_stat[key_count] = 1;
247                 print_key_found(puzzle, curr_key, key_comp[key_count], key_const);
248                 key_const = clearArr_2d(key_const);
249             } else {
250                 key_const = clearArr_2d(key_const);
251             }
252         }
253     }
254     else if (dir_state == 3) {
255         boundary_length = puzzle_cols - col ;
256         y_boundary = puzzle_rows - row;
257
258         if (y_boundary < boundary_length) {
259             boundary_length = y_boundary;
260         }
261         if (curr_key.length() <= boundary_length) {
262             finding_idx = 0;
263
264             while ((finding_idx < curr_key.length()) && (getCharFromString(curr_key, finding_idx) == puzzle[row+finding_idx][col+finding_idx])) {
265                 int[] key_idx = new int[2];
266                 key_idx[0] = row+finding_idx;
267                 key_idx[1] = col+finding_idx;
268                 key_const[finding_idx] = key_idx;
269                 key_comp[key_count]++;
270                 finding_idx++;
271             }
272             if (finding_idx == curr_key.length()) {
273                 keys_stat[key_count] = 1;
274                 print_key_found(puzzle, curr_key, key_comp[key_count], key_const);
275                 key_const = clearArr_2d(key_const);
276             }
277         }
278     }

```

```

276         } else {
277             key_const = clearArr_2d(key_const);
278         }
279     }
280 }
281 else if (dir_state == 4) {
282     boundary_length = puzzle_rows - row;
283
284     if (curr_key.length() <= boundary_length) {
285         finding_idx = 0;
286
287         while ((finding_idx < curr_key.length()) && (getCharFromString(curr_key, finding_idx) == puzzle[row+finding_idx][col])) {
288             int[] key_idx = new int[2];
289             key_idx[0] = row+finding_idx;
290             key_idx[1] = col;
291             key_const[finding_idx] = key_idx;
292             key_comp[key_count]++;
293             finding_idx++;
294         }
295         if (finding_idx == curr_key.length()) {
296             keys_stat[key_count] = 1;
297             print_key_found(puzzle, curr_key, key_comp[key_count], key_const);
298             key_const = clearArr_2d(key_const);
299         } else {
300             key_const = clearArr_2d(key_const);
301         }
302     }
303 }
304 else if (dir_state == 5) {
305     boundary_length = puzzle_rows - row;
306     x_boundary = col + 1;
307
308     if(x_boundary < boundary_length){
309         boundary_length = x_boundary;
310     }
311     if (curr_key.length() <= boundary_length) {
312         finding_idx = 0;
313
314         while ((finding_idx < curr_key.length()) && (getCharFromString(curr_key, finding_idx) == puzzle[row+finding_idx][col-finding_idx])) {
315             int[] key_idx = new int[2];
316             key_idx[0] = row+finding_idx;
317             key_idx[1] = col-finding_idx;
318             key_const[finding_idx] = key_idx;
319             key_comp[key_count]++;
320             finding_idx++;
321         }

```

```

322         if (finding_idx == curr_key.length()) {
323             keys_stat[key_count] = 1;
324             print_key_found(puzzle, curr_key, key_comp[key_count], key_const);
325             key_const = clearArr_2d(key_const);
326         } else {
327             key_const = clearArr_2d(key_const);
328         }
329     }
330 }
331 else if (dir_state == 6) {
332     boundary_length = col + 1;
333
334     if (curr_key.length() <= boundary_length) {
335         finding_idx = 0;
336
337         while ((finding_idx < curr_key.length()) && (getCharFromString(curr_key, finding_idx) == puzzle[row][col-finding_idx])) {
338             int[] key_idx = new int[2];
339             key_idx[0] = row;
340             key_idx[1] = col-finding_idx;
341             key_const[finding_idx] = key_idx;
342             key_comp[key_count]++;
343             finding_idx++;
344         }
345         if (finding_idx == curr_key.length()) {
346             keys_stat[key_count] = 1;
347             print_key_found(puzzle, curr_key, key_comp[key_count], key_const);
348             key_const = clearArr_2d(key_const);
349         } else {
350             key_const = clearArr_2d(key_const);
351         }
352     }
353 }
354 else if (dir_state == 7) {
355     boundary_length = row + 1;
356     x_boundary = col + 1;
357
358     if(x_boundary < boundary_length){
359         boundary_length = x_boundary;
360     }
361     if (curr_key.length() <= boundary_length) {
362         finding_idx = 0;
363
364         while ((finding_idx < curr_key.length()) && (getCharFromString(curr_key, finding_idx) == puzzle[row-finding_idx][col-finding_idx])) {
365             int[] key_idx = new int[2];
366             key_idx[0] = row-finding_idx;
367             key_idx[1] = col-finding_idx;

```

```

368         key_const[finding_idx] = key_idx;
369         key_comp[key_count]++;
370         finding_idx++;
371     }
372     if (finding_idx == curr_key.length()) {
373         keys_stat[key_count] = 1;
374         print_key_found(puzzle, curr_key, key_comp[key_count], key_const);
375         key_const = clearArr_2d(key_const);
376     } else {
377         key_const = clearArr_2d(key_const);
378     }
379 }
380 }
381 dir_state++;
382 }
383 col++;
384 }
385 row++;
386 }
387 if (isAllOne(keys_stat)) {
388     found = true;
389 }
390 key_count++;
391 }
392 long elapsedTime = System.nanoTime() - startTime;
393
394 // output
395 System.out.println("=====");
396 System.out.println("Execution time: " + (elapsedTime/1000000) + " ms");
397 System.out.println("=====");
398 }
399 }

```

Screenshot

Test Case 1

```
=====
(^///^) SELAMAT DATANG (^///^)
=====
Masukkan nama file (xxx.txt):
> Small1.txt
=====
```

[illegible]

```
=====
Execution time: 386 ms
=====
```

```
BAMBOOZLE
B - - - - - 
A - - - - - 
M - - - - - 
B - - - - - 
O - - - - - 
O - - - - - 
Z - - - - - 
L - - - - - 
E - - - - - 
- - - - - 
- - - - - 
- - - - - 
Total perbandingan huruf: 12

BEWILDERMENT
- - - - - 
- - - - - 
- - - - - 
- - - - - 
- - - - - 
- - - - - 
- - - - - 
- - - - - 
- - - - - 
- - - - - 
- - B E W I L D E R M E N T 
Total perbandingan huruf: 19
```

— — — — — B O U N C E — — — — —

— — — — — CAPETOWN — — — — —

— — — — C H A F E — — — —



WASNIAHC

A 10x10 grid of dots. In the fourth column from the left, the letters C, H, O, and W are arranged vertically, centered within the grid.

IF2211 | 14

DECREASE

- - - E S A E R C E D - - -
Total perbandingan huruf: 22

DUDS

- - - D U D S -
Total perbandingan huruf: 7

COMBAT

- - - T A B M O C - - -
Total perbandingan huruf: 60

CURIOUS

- - - C U R I O U S -
Total perbandingan huruf: 29

Test Case 2

(^///^) SELAMAT DATANG (^///^)

```
Masukkan nama file (xxx.txt):  
> Small2.txt
```

EMASCULATED

EMASCULATED - - -
Total perbandingan huruf: 47

FADES

Total perbandingan huruf: 9

Execution time: 737 ms

HEFTY

H	-	-	-	-
-	E	-	-	-
-	-	F	-	-
-	-	-	T	-
-	-	-	-	Y

Total perbandingan huruf: 18

INCITE

I - - - -
- N - - -
- - C - - -
- - - I - -
- - - - T -
- - - - - E

Total perbandingan huruf: 36

JACKET

J
A
C
K
E
T

Total perbandingan huruf: 8

KIDNAPPING

K
I
D
N
A
P
P
I
N
G

Total perbandingan huruf: 14

LIFTING

G
N
I
T
F
I
L

Total perbandingan huruf: 31

LOUVRE

E
R
V
U
O
L

Total perbandingan huruf: 53

[illegible]

MINES

OBFUSCATED

Total perbandingan huruf: 36

Test Case 3

[illegible][illegible][illegible]

Execution time: 536 ms

DEDICATION

DE
D
I
C
A
T
I
O
N

D
 E
 V
 E
 L
 O
 P

DRAFT

— T F A R D —

Total perbandingan huruf: 48

EARN

- - N - - - - -
- - - R - - - - -
- - - - A - - - -
- - - - - E - - - -

Total perbandingan huruf: 42

ELLIPTIC

- - - - - C -
- - - - - I -
- - - - - T -
- - - - - P -
- - - - - I -
- - - - - L -
- - - - - L -
- - - - - E -

Total perbandingan huruf: 60

EPHEMERAL

- - - - - L - - - -
- - - - - A - - - -
- - - - - R - - - -
- - - - - E - - - -
- - - - M - - - -
- - - E - - - -
- - H - - - -
- P - - - -
- E - - - -

Total perbandingan huruf: 62

FIASCO

- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- F I A S C O - - - -

Total perbandingan huruf: 32

Test Case 4

[illegible][illegible]

```
=====
Execution time: 1256 ms
=====
```

[illegible][illegible]

HEREDITARY

Y R A T I D E R E H

Total perbandingan huruf: 74

JACK

K
C
A
J

Total perbandingan huruf: 50

ICING

G
N
I
C
I

Total perbandingan huruf: 110

JOBS

S
B
O
J

Total perbandingan huruf: 82

MARRY

M
- A
- R
- R
- Y

Total perbandingan huruf: 80

MOTHER

- R E H T O M -

Total perbandingan huruf: 85

MIDDLEMAN

- M
- I
- D
- D
- L
- E
- M
- A
- N

Total perbandingan huruf: 16

NONSENSICAL

- L
- A
- C
- I
- S
- N
- E
- S
- N
- O
- N

Total perbandingan huruf: 65

Total perbandingan huruf: 13

Total perbandingan huruf: 16

Test Case 5

[illegible][illegible]

```
BASELINE  
- B -  
- A -  
- S -  
- E -  
- L -  
- I -  
- N -  
- E -  
  
Total perbandingan huruf: 9
```

BLAMELESS																			
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	B	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	L	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	M	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	E	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	L	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	E	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	S	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	S	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Total perbandingan huruf:										29									

```
=====
Execution time: 841 ms
=====
```

CLEANSHAVEN

- - N E V A H S N A E L C - -

Total perbandingan huruf: 52

CUSHION

- - - N -
- - - O -
- - - I -
- - H -
- S -
- U -
C -

Total perbandingan huruf: 40

COMBATING

- - - G -
- - - N -
- - - I -
- - - T -
- - - A -
- - - B -
- - - M -
- - - O -
- - - C -

Total perbandingan huruf: 68

DEMONSTRATE

- - - D E M O N S T R A T E - -

Total perbandingan huruf: 42

DRIBBLE

D

R

I

B

B

L

E

Total perbandingan huruf: 36

FLOGGING

G N I G G O L F

Total perbandingan huruf: 41

DUNGEON

D

U

N

G

E

O

N

Total perbandingan huruf: 26

GETTING

G

N

I

T

T

E

G

Total perbandingan huruf: 95

INCONVENIENT

IF2211 | 31

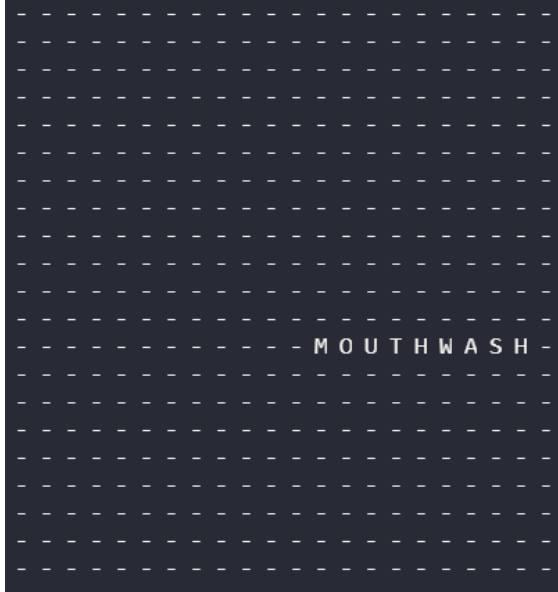
Test Case 6

```
=====
(^///^ ) SELAMAT DATANG (^///^)
=====
Masukkan nama file (xxx.txt):
> Med3.txt
=====
ITINERARY
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- - Y - - - - -
- - R - - - - -
- - A - - - - -
- - R - - - - -
- - E - - - - -
- - N - - - - -
- - I - - - - -
- - T - - - - -
- - I - - - - -
- - - - -
Total perbandingan huruf: 86
```

Execution time: 1276 ms

[illegible][illegible]

Total perbandingan huruf: 102



MOUTHWASH

```
Total perbandingan huruf: 75
```

OVERSHOT

Total perbandingan huruf: 8

Total perbandingan huruf: 81

F O O R P

```
RAVAGED  
- - - - -  
D - - - - -  
- E - - - - -  
- G - - - - -  
- A - - - - -  
- V - - - - -  
- A - - - - -  
- R - - - - -  
  
Total perbandingan huruf: 65
```

M
E
E
D
E
R

[illegible]

A 20x20 grid of dots on a dark blue background. The letters 'R', 'E', 'D', 'P', 'O', 'Y', 'E', and 'D' are placed at specific intersections of the grid lines, forming a sparse pattern.

Total perbandingan huruf: 59

DEPARCS

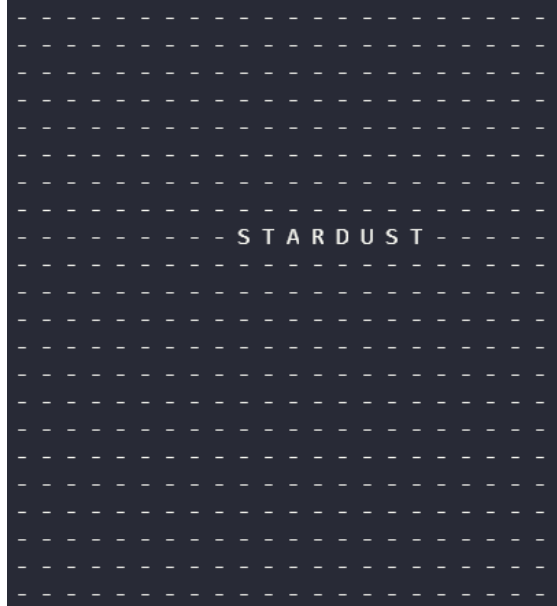
Total perbandingan huruf: 155

Total perbandingan huruf: 9

ROGUES

Total perbandingan huruf: 20

F
F
U
R
C
S



- - - - - STARDUST - - - - -

STUDENT
TESTED

SUPERVISORY

S
U
P
E
R
V
I
S
O
R
Y

Total perbandingan huruf: 31

Test Case 7

[illegible][illegible]

E
 R
 I
 P
 S
 A

Total perbandingan huruf: 43

BARGES

B
A
R
G
E
S

Total perbandingan huruf: 7

BLEAKLY

B
L
E
A
K
L
Y

Total perbandingan huruf: 38

[illegible][illegible]

[illegible]

Total perbandingan huruf: 190

[illegible]

Total perbandingan huruf: 118

Total perbandingan huruf: 21

[illegible][illegible]

Total perbandingan huruf: 29

DAKAR

D
A
K
A
R

Total perbandingan huruf: 211

DEMERIT

D
E
M
E
R
I
T

Total perbandingan huruf: 69

DOMINATED

- - - - - D E T A N I M O D - - - - -

Total perbandingan huruf: 47

ENTERTAIN

- - - - - N - - - - -

- - - - - I - - - - -

- - - - - A - - - - -

- - - - - T - - - - -

- - - - - R - - - - -

- - - - - E - - - - -

- - - - - T - - - - -

- - - - - N - - - - -

- - - - - E - - - - -

Total perbandingan huruf: 103



EXPLICABLE

FIST

IF2211 | 46

FORMALIST

Total perbandingan huruf: 75

A 20x20 grid of dots on a black background. The letters are placed at the following grid intersections (row, column) starting from the top-left:

- D: (15, 5)
- N: (14, 6)
- E: (13, 7)
- I: (12, 8)
- R: (11, 9)
- F: (10, 10)
- L: (9, 11)
- R: (8, 12)
- I: (7, 13)
- G: (6, 14)

Total perbandingan huruf: 75

[illegible]

HELM

M
L
E
H

Total perbandingan huruf: 55

[illegible][illegible][illegible]

```
=====
Execution time: 2181 ms
=====
```

Test Case 8

[illegible]

```
KEYWORD
- - - - - K
- - - - - E
- - - Y -
- - W -
- O -
R -
D -
```

Total perbandingan huruf: 43

LASSITUDE

EDUTISSAL

Total perbandingan huruf: 169

```
LINGUIST
```

```
      T
    - S
   - I
  - U
 - G
N
- I
 L
```

Total perbandingan huruf: 135

METASTASIS

M
E
T
A
S
T
A
S
I
S

Total perbandingan huruf: 68

NUMERATE

N
U
M
E
R
A
T
E

Total perbandingan huruf: 15

OVERDRIVE

O
V
E
R
D
R
I
V
E

Total perbandingan huruf: 105

Total perbandingan huruf: 105

OVERPAYMENT

T N E M Y A P R E V O

Total perbandingan huruf: 73

Total perbandingan huruf: 73

Total perbandingan huruf: 186

Y
L
D
E
T
N
I
O
P

Total perbandingan huruf: 61

[illegible]

Total perbandingan huruf: 46

PROSPERED

P
R
O
S
P
E
R
E
D

Total perbandingan huruf: 126

Total perbandingan huruf: 126

[illegible]

RHEUMATIC

Total perbandingan huruf: 52

[illegible]

Total perbandingan huruf: 145

SEEPAGE

E
G
A
P
E
E
S

Total perbandingan huruf: 223

Total perbandingan huruf: 223

SKETCHPAD

D
 A
 P
 H
 C
 T
 E
 K
 S

Total perbandingan huruf: 206

SKINNING

G
 N
 I
 N
 N
 I
 K
 S

Total perbandingan huruf: 184

STARLING

- - - - - S T A R L I N G - - - - -

Total perbandingan huruf: 88

[illegible]

TIMETABLE

[illegible]

Total perbandingan huruf: 120

TINKERING

G N I R E K N I T

Total perbandingan huruf: 49

UNBIASED

D E S A I B N U

Total perbandingan huruf: 86

ZIGZAG

105

Total perbandingan huruf: 105

Execution time: 3126 ms

Alamat Drive

<https://drive.google.com/drive/folders/1DG3KXAZH4BZwTDxs3RHe10UGzy8ntT-0?usp=sharing>

[sivaren/Stima-Tucil1: Penyelesaian Word Search Puzzle dengan Algoritma Brute Force \(github.com\)](#)

Lampiran

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan (no syntax error)	√	
2. Program berhasil <i>running</i>	√	
3. Program dapat membaca file masukan dan menuliskan luaran.	√	
4. Program berhasil menemukan semua kata didalam puzzle.	√	

s